Bounding Resource Consumption with Gödel-Dummett Logics

Dominique Larchey-Wendling

Bounding resource consumption with Gödel-Dummett logics

Dominique Larchey-Wendling

LORIA – CNRS
Campus scientifique, BP 239
54 506 Vandœuvre-lès-Nancy, France

Abstract. Gödel-Dummett logic LC and its finite approximations LC_n are the intermediate logics complete w.r.t. linearly ordered Kripke models. In this paper, we use LC_n logics as a tool to bound resource consumption in some process calculi. We introduce a non-deterministic process calculus where the consumption of a particular resource denoted $\bullet$ is explicit and provide an operational semantics which measures the consumption of this resource. We present a linear transformation of a process P into a formula f of LC. We show that the consumption of the resource by P can be bounded by the positive integer n if and only if the formula f admits a counter-model in LC_n. Combining this result with our previous results on proof and counter-model construction for LC_n, we conclude that bounding resource consumption is (linearly) equivalent to searching counter-models in LC_n.

1 Introduction

Gödel-Dummett logic LC and its finitary versions $(\mathsf{LC}_n)_{n>0}$ are the intermediate logics (between classical logic CL and intuitionistic logic IL) characterized by linear Kripke models. LC was introduced by Gödel in [1] and later axiomatized by Dummett in [2]. It is now one of the most studied intermediate logics and has been recognized recently as one of the fundamental *t-norm based fuzzy logics* [3]. Proof-search in LC has benefited from the development of proof-search in intuitionistic logic IL with two important seeds: the *contraction-free calculus* of Dyckhoff [4–6] and the *hyper-sequent* calculus of Avron [7,8]. Two recent contributions propose an alternative approach based on a set of *local* and *strongly invertible* proof rules (for either sequent [9] or hyper-sequent [7,10] calculus) and a semantic criterion to decide *irreducible (hyper)-sequents* and eventually build a counter-model.

In our previous work, we proposed a new approach to the decision problem in LC [11]. We transform a formula (or a sequent) of LC into a *conditional bi-colored graph* of the same size. Then, we search counter-models of the initial formula by looking for chains of a certain kind in the graph. We call those chains *alternating chains*. This method constitutes a decision procedure for LC and LC_n, thus we have a linear transformation of the decision problem for LC (and also LC_n) into the search for alternating chains problem in conditional graphs. Moreover,

we propose a procedure based on matrix computation to solve the search of alternating chains [12].

In this paper, we study the reverse transformation. First, we characterize the search for alternating chains in conditional graphs as a resource use bounding problem in some particular process calculus: the processes have *nondeterministic* branching, *conditional* branching, *consume* some particular resource, and can be *recursive*. The conditions are expressed by boolean formulae. We show how to normalize process systems being thus able to view theses processes as conditional graphs. We relate resource consumption by processes to the search for alternating chains in conditional graphs.

Then we show how to encode a conditional graph into a formula of LC (of size linear w.r.t. the size of the graph). We prove the equivalence of the existence of a counter-model for this formula and the existence of an alternating chain into the conditional graph. Therefore we obtain a linear transformation of the search for alternating chains problem in conditional graphs into a decision problem for the family of logics LC_n.

Moreover, this result establishes a characterization of the family of Gödel-Dummett logics LC_n as resource use bounding logics for some particular process calculus. This is the main goal of this paper: to shed some new light of LC and to relate it with processes and resource consumption. In particular, the process calculus we introduce here should not be viewed as a real tool to model complex systems. However, it could be extended or integrated to other process calculi like CCS [13] so as to exploit the LC logic for specifying resource related properties.

2 Logical syntax for CL and LC

In this section we present the syntax we use for logical formulae. We either use formulae of classical propositional logic CL or formulae of propositional Gödel-Dummett logic LC. Fortunately they share the same syntax, even though their semantics differ.

The set of propositional *formulae*, denoted Form, is defined inductively, starting from a set of propositional *variables*, denoted by LVar, with an additional bottom constant $\bot$ denoting *absurdity* and using the connectives $\wedge$, $\vee$ and $\supset$:

$$\mathsf{Form}: \quad f ::= \bot \mid x \mid f \wedge f \mid f \vee f \mid f \supset f \quad \text{where } x \in \mathsf{LVar}$$

We use the abbreviations $\neg f \equiv f \supset \bot$ and $\top \equiv \neg\bot$. In conditional graphs (see sections 4 and 6), we even use the notation $\overline{x} \equiv \neg x$ but only for propositional variables. The semantics of classical formulae is as usual: given a *valuation* (or interpretation) of propositional variables $\sigma : \mathsf{LVar} \to \{0,1\}$, the semantic value of f denoted by $[\![f]\!]_\sigma \in \{0,1\}$ is defined inductively on the structure of f, the connectives $\bot$, $\supset$, $\wedge$ and $\vee$ being respectively interpreted by their corresponding boolean operator. The semantics of the formulae of LC will be defined later in section 5.1.

3 Resource consuming processes

In this section, we present a calculus for processes which consume resources. This calculus features non-determinism, choices, recursion and of course resource consumption. We only model one kind of resources in our calculus, denoted by a big dot $\bullet$. The processes only consume resources, they do not produce them, neither do they transform them into another kind of resources. So the behavior of processes is characterized by how they consume resources.

Let us consider a set of *process variables* denoted PVar. We define the set of *processes* Proc as follows:

$$\mathsf{Proc}: \quad P ::= X \mid <f> P \mid \bullet P \mid [P + \cdots + P] \quad \text{where } f \in \mathsf{Form}, X \in \mathsf{PVar}$$

The process $\bullet P$ should be viewed as the process which consumes one resource $\bullet$ and then, behaves as P. The process $[P + Q]$ should be understood as the non-deterministic sum of P and Q, i.e. the process that behaves either as P or Q. In particular, the process $[\,]$ does nothing, i.e. does not consume any resource. Given a boolean condition f, the process $<f> P$ is the process which behaves like P if the condition f is fulfilled, and does nothing otherwise.

Non-determinism and conditions are sufficient to represent the if-then-else construct: if f then P else $Q \equiv [<f> P + <\neg f> Q]$. We insist on the fact that the boolean value of the condition f does not evolve during the execution of processes: it is fixed once and for all before the execution starts. And as we are going to describe the operational semantics of processes, it should be noted that conditions (like f) are *external*: even though they influence the behavior of processes, they cannot change because of that behavior.

3.1 Process systems and recursion

To represent recursion, i.e. the possibility for processes to become themselves again after having consumed some resources, we use (sets of) recursive equations:

Definition 1 (Process system). *A* process system *is a pair* $(\mathcal{E}, \mathcal{V})$ *where* $\mathcal{E} = \{X_1 = P_1, \ldots, X_n = P_n\}$ *is a finite* set of process equations. $X_1, \ldots, X_n$ *are supposed are to be* n *distinct process variables and* $\mathcal{V} \subseteq \{X_1, \ldots, X_n\}$ *is a subset of* relevant variables. $P_1, \ldots, P_n$ *are processes. The variables occurring in* $\mathcal{E}$ *but not in* $\mathcal{V}$ *are called* private *variables. A* sub-process *of* $\mathcal{E}$ *is either one of* $X_1, \ldots, X_n$ *or a sub-term of one of* $P_1, \ldots, P_n$.

As an example, consider the system

$$\mathcal{V} = \{X\} \qquad \mathcal{E} = \begin{cases} X = \bullet <a> Y \\ Y = [<b> X + <\neg a> \bullet Z + Z] \\ Z = [\,] \end{cases}$$

$$[<b> X + <\neg a> \bullet Z + Z]$$

X	$<b> X$	$[\,]$
Y	$<a> Y$	$\bullet <a> Y$
Z	$\bullet Z$	$<\neg a> \bullet Z$

The sub-processes are listed on the right-hand side. Intuitively, X is the process which consumes one resource and if a is true becomes Y. Y becomes either X

$$\frac{}{P \multimap[0,\sigma,\mathcal{E}]\bullet P}\ [\text{Id}] \qquad \frac{P_i \multimap[n,\sigma,\mathcal{E}]\bullet Q}{[\cdots + P_i + \cdots] \multimap[n,\sigma,\mathcal{E}]\bullet Q}\ [\text{Sum}] \qquad \frac{P \multimap[n,\sigma,\mathcal{E}]\bullet Q}{\bullet P \multimap[n+1,\sigma,\mathcal{E}]\bullet Q}\ [\text{Res}]$$

$$\frac{P \multimap[n,\sigma,\mathcal{E}]\bullet Q \qquad X = P \in \mathcal{E}}{X \multimap[n,\sigma,\mathcal{E}]\bullet Q}\ [\text{Eq}] \qquad \frac{P \multimap[n,\sigma,\mathcal{E}]\bullet Q \qquad [\![f]\!]_\sigma = 1}{<f> P \multimap[n,\sigma,\mathcal{E}]\bullet Q}\ [\text{Con}]$$

Fig. 1. Deduction system for resource consumption

if b is true, or if a is false consumes one resource and then becomes Z, or Y becomes Z. Z is the process that does nothing.

This calculus should not be viewed as useful for representing real or complex systems. It has too few features for that. But it could be viewed as an *abstraction* calculus: either one could abstract a complex system into our simple formalism to prove resource consumption related properties for this particular system, or one could extend our calculus with further constructs to model more sophisticated systems directly.

3.2 Operational semantics for resource consumption

Given a set of process equations $\mathcal{E}$ and a valuation $\sigma : \mathsf{LVar} \to \{0,1\}$ of boolean variables, we define the ternary relation $\multimap[\cdot,\sigma,\mathcal{E}]\bullet \subseteq \mathsf{Proc} \times \mathbb{N} \times \mathsf{Proc}$ by the set of deduction rules presented in figure 1. As the reader might notice, σ and $\mathcal{E}$ are not modified by the application of those rules but they occur in the side conditions of rules [Eq] and [Con], restricting the applicability of those rules. When $\mathcal{E}$ or σ are understood in the context, we might simplify the notation $P \multimap[n,\sigma,\mathcal{E}]\bullet Q$ into $P \multimap[n,\sigma]\bullet Q$ or even $P \multimap[n]\bullet Q$. Intuitively, $P \multimap[n]\bullet Q$ should be read as: the process P has an execution path to Q which consumes exactly n times the resource $\bullet$.

Lemma 1. *The* [*Cut*] *rule is admissible:*

$$\frac{P \multimap[m]\bullet Q \qquad Q \multimap[n]\bullet R}{P \multimap[m+n]\bullet R}\ [\mathit{Cut}]$$

Proof. We prove the result by induction on the length of the deduction of $P \multimap[m]\bullet Q$. If $P \multimap[m]\bullet Q$ is obtained by the axiom [Id], then $Q \equiv P$ and $m = 0$, thus $P \multimap[0+n]\bullet R$ is identical to $Q \multimap[n]\bullet R$. If $P \multimap[m]\bullet Q$ is obtained by the [Res] rule then $P \equiv \bullet P'$ and we have a shorter (sub-)deduction of $P' \multimap[m-1]\bullet Q$. By induction, $P' \multimap[m-1+n]\bullet R$ is deducible and then, applying rule [Res], we obtain $\bullet P' \multimap[m-1+n+1]\bullet R$, thus $P \multimap[m+n]\bullet R$. If the last rule is [Eq], then $P \equiv X$ with $X = P' \in \mathcal{E}$ and we have a sub-deduction of $P' \multimap[m]\bullet Q$. Thus, by induction, we obtain a deduction of $P' \multimap[m+n]\bullet R$.

Applying rule [Eq], we obtain a deduction of $P \dashv[m+n]\vdash\bullet R$. The cases of rules [Sum] and [Con] are similar. □

Definition 2 (Boundable resource use). *A process system $(\mathcal{E}, \mathcal{V})$ has* a resource use boundable by n *if there exists a valuation $\sigma : \mathsf{LVar} \to \{0,1\}$ such that for any $X, Y \in \mathcal{V}$ and $k \in \mathbb{N}$, if $X \dashv[k, \sigma, \mathcal{E}]\vdash\bullet Y$ holds then $k \leqslant n$.*

This definition means that the resource use *can be bounded* in some context, modeled by the valuation σ. The resource use is not necessarily bounded in every context.

3.3 Normalization

We define the equivalence of process systems and a normalization procedure so that the systems appear in a shape suitable for further transformations. The process systems $(\mathcal{E}, \mathcal{V})$ and $(\mathcal{F}, \mathcal{V})$ are *equivalent* if for any valuation σ, the relations $\dashv[\cdot, \sigma, \mathcal{E}]\vdash\bullet$ and $\dashv[\cdot, \sigma, \mathcal{F}]\vdash\bullet$ have identical restrictions to $\mathcal{V} \times \mathbb{N} \times \mathcal{V}$.

Definition 3 (Normality). *A process equation is* normal *if it contains no nested construct, i.e. it has one of the following forms: $X = \bullet Y$, $X = <f> Y$ or $X = [Y_1 + \cdots + Y_n]$ where Y and the Y_i's are process variables. A process system is* normal *if all its equations are normal.*

Lemma 2 (Normalization). *Let $(\mathcal{E}, \mathcal{V})$ be a process system of size k. There exists a normal process system $(\mathcal{E}', \mathcal{V})$ of size $\mathcal{O}(k)$ which is equivalent to $(\mathcal{E}, \mathcal{V})$.*

Proof. Let $(\mathcal{E}, \mathcal{V})$ be a process system of size k. We build the set of equations of the system $\mathcal{E}'$. Let us introduce a new process variable X_P for each strict (i.e. not a process variable) sub-process P of $\mathcal{E}$. For process variables $P \equiv Y$ occurring in $\mathcal{E}$, we choose $X_Y \equiv Y$, so there is no new process variable for atomic sub-processes. Let P be a sub-process of $\mathcal{E}$. If $P \equiv \bullet Q$, we add the equation $X_P = \bullet X_Q$ to $\mathcal{E}'$. If $P \equiv <f> Q$, we add $X_P = <f> X_Q$ and if $P \equiv [Q_1 + \cdots + Q_k]$, we add $X_P = [X_{Q_1} + \cdots + X_{Q_k}]$. Finally, if $Y = P$ is an equation of $\mathcal{E}$, we add the equation $Y = [X_P]$ to $\mathcal{E}'$.

Obviously $(\mathcal{E}', \mathcal{V})$ is a normal system and its size (number of symbols) is linear in the size of $(\mathcal{E}, \mathcal{V})$. It is a bit tedious but obvious to show that for any sub-process $P, Q \in \mathcal{E}$, $\sigma : \mathsf{LVar} \to \{0,1\}$ and $n \in \mathbb{N}$, $P \dashv[n, \sigma, \mathcal{E}]\vdash\bullet Q$ holds if and only if $X_P \dashv[n, \sigma, \mathcal{E}']\vdash\bullet X_Q$ holds. The proof can be done by induction on the length of deductions. Then, since for any variable Y of $\mathcal{V}$ we have the property $X_Y \equiv Y$, the relation $\dashv[\cdot, \sigma, \mathcal{E}]\vdash\bullet$ and $\dashv[\cdot, \sigma, \mathcal{E}']\vdash\bullet$ have identical restrictions to $\mathcal{V} \times \mathbb{N} \times \mathcal{V}$. □

The result of the normalization procedure described previously applied to the example presented in section 3.1 is the following:

$$\begin{array}{llllll} X = [K_6] & Z = [K_5] & K_2 = <b> X & K_4 = \bullet Z & K_1 = [K_2 + K_7 + Z] \\ Y = [K_1] & K_6 = \bullet K_3 & K_3 = <a> Y & K_5 = [\,] & K_7 = <\neg a> K_4 \end{array}$$

4 Conditional bi-colored graphs

In this section, we introduce the notion of *conditional graphs*. Then we show how to transform a normal process system into a conditional graph and the relation between the existence of some chains in those graphs and the operational semantics of the process system.

4.1 Graphs and instance graphs

A *bi-colored graph* is a (finite) directed graph with two kinds of arrows: *green* arrows denoted by $\rightarrow$ and *red* arrows denoted by $\Rightarrow$.

Definition 4 (alternating chain). *A* n-alternating chain *is a chain of the form* $(\rightarrow^\star\Rightarrow)^n$.

So a chain contains a n-alternating chain if and only if it contains at least n red arrows $\Rightarrow$.

Definition 5 (Conditional graph). *A* conditional bi-colored graph *is a bi-colored graph where green arrows* $\rightarrow$ *may be indexed with the (propositional) boolean expressions of* Form.

Considering boolean expressions as representatives for boolean functions and given a valuation $\sigma : \mathsf{LVar} \rightarrow \{0,1\}$, a boolean expression e is instantiated to the boolean value $[\![e]\!]_\sigma \in \{0,1\}$. We obtain an instance graph: an arrow indexed with a boolean expression e belongs to this instance if and only if $[\![e]\!]_\sigma = 1$. The case of an unconditional (i.e. not indexed) arrow can be treated by considering that it has an implicit boolean conditional which is the tautology $\top$ (and then always evaluates to 1) and non-existing arrows have the implicit boolean condition $\bot$ that always evaluates to 0.

Definition 6 (Instance graph). *Let* $\mathcal{G}$ *be a conditional bi-colored graph and* σ *be a valuation of boolean variables in* $\{0,1\}$. *We define the* instance graph $\mathcal{G}_\sigma$ *as the bi-colored graph that one obtains when one evaluates boolean expressions indexing arrows and keeping exactly those whose valuation equals* 1.

4.2 From normal process systems to conditional graphs

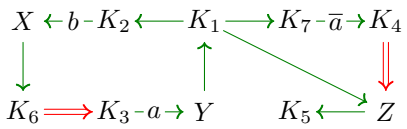
We describe how to build a conditional bi-colored graph from a normal set of process equations. Let $\mathcal{E}$ be a finite set of normal process equations. The equations are of one of the following forms: $X = \bullet Y$, $X = <f>Y$ or $X = [Y_1 + \cdots + Y_k]$ where Y and the Y_i's are process variables.

We build the graph $\mathcal{G}_\mathcal{E}$ (simply denoted $\mathcal{G}$ here). It has the process variables occurring in $\mathcal{E}$ as vertices. We associate to each (normal) equation of $\mathcal{E}$ a set of arrows:

- for the equation $X = \bullet Y$, we associate the arrow $X \Rightarrow Y$;

– for $X = <f>Y$, we associate the arrow $X \rightarrow_f Y$;
– for $X = [Y_1 + \cdots + Y_k]$, we associate the arrows $X \rightarrow Y_1, \ldots, X \rightarrow Y_k$.

Obviously, the graph $\mathcal{G} \equiv \mathcal{G}_\mathcal{E}$ is a conditional bi-colored graph and its size is linear in the size of $\mathcal{E}$. As an example, we display the graph associated with the example of normal process system obtained in section 3.3:



4.3 Chains and resource consumption

In this section, we relate the consumption of the resource $\bullet$ by the processes of $\mathcal{E}$ to the alternating chains of $\mathcal{G}_\mathcal{E}$.

Theorem 1. *Let $\mathcal{E}$ be a normal set of process equations and $\mathcal{G} = \mathcal{G}_\mathcal{E}$ be its associated conditional graph. Let $\sigma : \mathsf{LVar} \rightarrow \{0,1\}$ be a valuation. Then for any process variables $X, Y \in \mathcal{E}$ and any $n \in \mathbb{N}$, $X \dashv[n,\sigma]\vdash\bullet\ Y$ holds if and only if there exists a chain $X\ (\rightarrow^\star\Rightarrow)^n\rightarrow^\star Y$ in the instance graph $\mathcal{G}_\sigma$.*

Proof. Let us fix $\mathcal{E}$ and $\sigma : \mathsf{LVar} \rightarrow \{0,1\}$. We consider the conditional graph $\mathcal{G}_\mathcal{E}$ and its instance $\mathcal{G}_\sigma$. Let $X \Rightarrow Y$ be an arrow of $\mathcal{G}_\sigma$. By construction of $\mathcal{G}_\mathcal{E}$, there is an equation $X = \bullet Y$ in $\mathcal{E}$. Thus, $X \dashv[1]\vdash\bullet\ Y$ holds. Now let us consider an arrow $X \rightarrow Y$ of $\mathcal{G}_\sigma$: either there exists an equation $X = <f>Y$ in $\mathcal{E}$ s.t. $[\![f]\!]_\sigma = 1$ or there exists an equation $X = [\cdots + Y + \cdots]$ in $\mathcal{E}$. In either case, $X \dashv[0]\vdash\bullet\ Y$ holds. Then by using the derived [Cut] rule of lemma 1, from a chain $X\ (\rightarrow^\star\Rightarrow)^n\rightarrow^\star Y$ containing exactly n red arrows $\Rightarrow$, we can deduce that $X \dashv[n]\vdash\bullet\ Y$ holds.

Conversely, let us show how to transform a deduction of $X \dashv[n]\vdash\bullet\ Y$ into a chain of the form $X\ (\rightarrow^\star\Rightarrow)^n\rightarrow^\star Y$ in $\mathcal{G}_\sigma$, by induction on the length of the deduction. Suppose it ends with the [Id] rule. Then $Y \equiv X$ and $n = 0$. Thus $X \rightarrow^0 Y$ is a zero length chain. Considering other rules, we remark that a deduction of $X \dashv[n]\vdash\bullet\ Y$ cannot end with rules [Sum], [Res] or [Con] since X is a process variable. We consider the last remaining case where the deduction ends with rule [Eq]. Then there exists an equation $X = P$ in $\mathcal{E}$. As $\mathcal{E}$ is normal, P is in one of the following forms: $\bullet Z$, $<f>Z$ or $[Z_1 + \cdots + Z_k]$, where Z and the Z_i are process variables:

– if $P \equiv \bullet Z$ then there is a (sub-)deduction of $\bullet Z \dashv[n]\vdash\bullet\ Y$ and therefore a (sub-)deduction of $Z \dashv[n-1]\vdash\bullet\ Y$. By induction on deductions, we obtain of chain $Z\ (\rightarrow^\star\Rightarrow)^{n-1}\rightarrow^\star Y$ in $\mathcal{G}_\sigma$. Moreover, as $X = \bullet Z \in \mathcal{E}$, we have an arrow $X \Rightarrow Z$ in $\mathcal{G}_\sigma$. Thus there exists a chain $X\ (\rightarrow^\star\Rightarrow)^n\rightarrow^\star Y$ in $\mathcal{G}_\sigma$;
– if $P \equiv <f>Z$, then we have a sub-deduction of $<f>Z \dashv[n]\vdash\bullet\ Y$. It is necessary that the last rule of this sub-deduction is [Cond] and then $[\![f]\!]_\sigma = 1$. We have a sub-deduction of $Z \dashv[n]\vdash\bullet\ Y$. By induction, there is a chain $Z\ (\rightarrow^\star\Rightarrow)^n\rightarrow^\star Y$ in $\mathcal{G}_\sigma$. Since $X = <f>Z \in \mathcal{E}$ and $[\![f]\!]_\sigma = 1$, there is an arrow $X \rightarrow Z$ in $\mathcal{G}_\sigma$. Thus there exists a chain $X\ (\rightarrow^\star\Rightarrow)^n\rightarrow^\star Y$ in $\mathcal{G}_\sigma$;

– if $P \equiv [Z_1 + \cdots + Z_k]$, then we have a sub-deduction of $[Z_1 + \cdots + Z_k] \dashv\!\lfloor n \rfloor\!\vdash\bullet\, Y$. It is necessary that the last rule of this sub-deduction is [Sum] and thus, there exists $i \in [1, k]$ and a sub-deduction of $Z_i \dashv\!\lfloor n \rfloor\!\vdash\bullet\, Y$. By induction, we obtain a chain $Z_i\ (\rightarrow^\star\Rightarrow)^n\rightarrow^\star Y$. Since $X = [Z_1 + \cdots + Z_k] \in \mathcal{E}$, there is an arrow $X \rightarrow Z_i$ in $\mathcal{G}_\sigma$, and there exists a chain $X\ (\rightarrow^\star\Rightarrow)^n\rightarrow^\star Y$ in $\mathcal{G}_\sigma$. □

Corollary 1. *Let $\mathcal{E}$ be a normal set of process equations, $\mathcal{V}$ the set of variables of $\mathcal{E}$, $\mathcal{G}$ its associated conditional graph and $n \in \mathbb{N}$. The process system $(\mathcal{E}, \mathcal{V})$ has a resource use boundable by n if and only if there is an instance graph $\mathcal{G}_\sigma$ with no $(n+1)$-alternating chain.*

Proof. If $(\mathcal{E}, \mathcal{V})$ has a resource use boundable by n, there exists a valuation $\sigma : \mathsf{LVar} \rightarrow \{0, 1\}$ s.t. for any $X, Y \in \mathcal{V}$ and $k \in \mathbb{N}$, if $X \dashv\!\lfloor k, \sigma \rfloor\!\vdash\bullet\, Y$ holds then $k \leqslant n$. Suppose there exists a $(n+1)$-alternating chain $X_0\ (\rightarrow^\star\Rightarrow)^{n+1}\ Y_0$ in the instance graph $\mathcal{G}_\sigma$. Then, by theorem 1, $X_0 \dashv\!\lfloor n+1, \sigma \rfloor\!\vdash\bullet\, Y_0$. But $X_0, Y_0 \in \mathcal{V}$ so we get $n + 1 \leqslant n$, that leads to a contradiction. Conversely, let $X_0, Y_0 \in \mathcal{V}$ and $k \in \mathbb{N}$ satisfying $X_0 \dashv\!\lfloor k, \sigma \rfloor\!\vdash\bullet\, Y_0$. Then, by theorem 1, there is a chain $X_0\ (\rightarrow^\star\Rightarrow)^k\rightarrow^\star Y_0$ in the instance graph $\mathcal{G}_\sigma$. If $k > n$ then this chain contains a sub-chain of the form $X_0\ (\rightarrow^\star\Rightarrow)^{n+1}\ Y_1$, contradiction. Consequently $k \leqslant n$. □

4.4 From conditional graphs to normal process systems

We have proved that a normal process system can be transformed into a conditional graph of the same size. Now we present the converse transformation. We show how to recover a process system from a conditional graph.

There is only a slight problem to be addressed: the construction described in section 4.2 does not generate a configuration like for example $X \rightarrow Y$ and $X \Rightarrow Z$ where these two arrows have the same source X. This can only be generated when these two arrows are green $\rightarrow$ using the $[\cdots + \cdots]$ construct. To overcome this difficulty, we propose the following trick: every red arrows $\Rightarrow$ (resp. conditional arrow $\rightarrow_f$) is split into two arrows $\rightarrow\Rightarrow$ (resp. $\rightarrow\rightarrow_f$) introducing a new intermediary node for each red $\Rightarrow$ and conditional $\rightarrow_f$ arrow. These two splits preserve n-alternating chains.

Using such a transformation on a conditional graph $\mathcal{G}$, we obtain a new conditional graph $\mathcal{G}'$ with the following property: every node which is the source of multiple arrows is the source of only unconditional green arrows $\rightarrow$ since the source of a red $\Rightarrow$ (resp. conditional $\rightarrow_f$) arrow is the intermediary node which is specifically introduced for this particular arrow.

The graph $\mathcal{G}'$ can be transformed into a normal set of process equations. Let $\mathcal{V}$ be a set of process variables, one X_v for each vertex v of $\mathcal{G}'$. We build $\mathcal{E}$ as follows. We consider each vertex as a source for some arrows and associate an equation to each vertex:

– if v is the source of a red arrow $v \Rightarrow w$ then v is the source of no other arrow[1] and we add $X_v = \bullet\, X_w$ to $\mathcal{E}$;

[1] v is new and has been introduced in $\mathcal{G}'$ specifically for this purpose.

– if v is the source of a conditional green arrow $v \rightarrow_f w$ then v is the source of no other arrow and we add $X_v = <f> X_w$ to $\mathcal{E}$;
– otherwise v is the source of k (unconditional) green arrows $v \rightarrow w_1, \ldots, v \rightarrow w_k$ (k could be 0) and we add $X_v = [X_{w_1} + \cdots + X_{w_k}]$ to $\mathcal{E}$.

$\mathcal{E}$ being built this way, it is obvious that it is a normal set of process equations and that $\mathcal{G}'$ appears as the conditional graph associated with $\mathcal{E}$ by the construction described in section 4.2, i.e. $\mathcal{G}' = \mathcal{G}_{\mathcal{E}}$.

Theorem 2. *Let $\mathcal{G}$ be a conditional graph of size k. There exists a process system $(\mathcal{E}, \mathcal{V})$ of size $\mathcal{O}(k)$ with the following property: for any $n \in \mathbb{N}$, $(\mathcal{E}, \mathcal{V})$ has a resource use boundable by n if and only if there is an instance graph $\mathcal{G}_\sigma$ with no $(n+1)$-alternating chain.*

Proof. Obviously, the size of the graph $\mathcal{G}'$ described earlier in this section is less than $2k$. The size of $\mathcal{E}$ is the same as the size of $\mathcal{G}'$, so is less than $2k$. Let $\mathcal{V}$ be the set of process variables occurring in $\mathcal{E}$. The size of $(\mathcal{E}, \mathcal{V})$ is less than $3k$. Then it is clear that for any $\sigma : \mathsf{LVar} \rightarrow \{0, 1\}$, there is a one-to-one correspondence between a $(n+1)$-alternating chain of $\mathcal{G}_\sigma$ and a $(n+1)$-alternating chain of $\mathcal{G}'_\sigma$, since the splits $\Rightarrow \rightsquigarrow \rightarrow\Rightarrow$ and $\rightarrow_f \rightsquigarrow \rightarrow\rightarrow_f$ preserve alternating chains in every instance. As the identity $\mathcal{G}' = \mathcal{G}_{\mathcal{E}}$ holds, we finish by an application of corollary 1 to $\mathcal{E}$. □

5 From conditional graphs to LC

In this section, we introduce the algebraic semantics of the family of propositional Gödel-Dummett logics LC_n. The value n belongs to the set $\overline{\mathbb{N}}^\star = \{1, 2, \ldots\} \cup \{\infty\}$ of strictly positive natural numbers with its natural order $\leqslant$, augmented with a greatest element ∞. In the case $n = \infty$, the logic LC_∞ is also denoted by LC: this is the usual Gödel-Dummett logic.

After having defined the semantics of LC_n, we show how to transform a conditional graph into a formula[2] of LC_n and we relate the existence of a counter-model for this formula to the alternating chains of the initial graph, and thus to resource consumption.

5.1 The semantics of LC_n

IL denotes the set of formulae that are provable in any intuitionistic propositional calculus (see [4]) and CL denotes the classically valid formulae. As usual an *intermediate propositional logic* [6] is a set of formulae $\mathcal{L}$ satisfying $\mathsf{IL} \subseteq \mathcal{L} \subseteq \mathsf{CL}$ and closed under the rule of modus ponens (if $A \in \mathcal{L}$ and $A \supset B \in \mathcal{L}$ then $B \in \mathcal{L}$) and under arbitrary substitution (if $A \in \mathcal{L}$ and ρ is any substitution then $A_\rho \in \mathcal{L}$.)

[2] In fact, into a sequent which can straightforwardly be transformed into an equivalent formula, see proposition 1.

For any $n \in \overline{\mathbb{N}}^\star$, the Gödel-Dummett logic LC_n is an intermediate logic. On the semantic side, it is characterized by the linear Kripke models of size n, see [2]. The following strictly increasing sequence holds:

$$\mathsf{IL} \subset \mathsf{LC} = \mathsf{LC}_\infty \subset \cdots \subset \mathsf{LC}_n \subset \cdots \subset \mathsf{LC}_1 = \mathsf{CL}$$

In the particular case of LC, the logic has a simple Hilbert axiomatic system: $(X \supset Y) \vee (Y \supset X)$ added to the axioms of IL.

In this paper, we will use the algebraic semantic characterization of LC_n [7] rather than the Kripke semantics. Let us fix a particular $n \in \overline{\mathbb{N}}^\star$. The algebraic model is the set $\overline{[0,n)} = [0,\ldots,n[\cup \{\infty\}$ composed of $n+1$ elements.[3] A valuation (or interpretation) on propositional variables $\sigma : \mathsf{LVar} \to \overline{[0,n)}$ is inductively extended to formulae:

$$\begin{array}{lll} [\![\bot]\!]_\sigma = 0 & [\![a \vee b]\!]_\sigma = \max(a,b) & \\ [\![x]\!]_\sigma = \sigma_x & [\![a \wedge b]\!]_\sigma = \min(a,b) & [\![a \supset b]\!]_\sigma = a \twoheadrightarrow b \end{array}$$

where the operator $\twoheadrightarrow$ is defined by $a \twoheadrightarrow b =$ if $a \leqslant b$ then ∞ else b. A formula f is *valid* for the valuation σ if the equality $[\![f]\!]_\sigma = \infty$ holds. This interpretation is complete for LC_n. A *counter-model* of a formula f is a valuation σ such that $[\![f]\!] < \infty$.

A *sequent* is a pair $\Gamma \vdash \Delta$ where Γ and Δ are multisets of formulae. Γ, Δ denotes the sum of the two multisets and if Γ is the empty multiset, we write $\vdash \Delta$. Given a sequent $\Gamma \vdash \Delta$ and an interpretation $[\![\cdot]\!]$ of variables, we interpret $\Gamma \equiv a_1, \ldots, a_n$ by $\llbracket \Gamma \rrbracket = \min\{[\![a_1]\!], \ldots, [\![a_n]\!]\}$ and $\Delta \equiv b_1, \ldots, b_p$ by $\llbracket \Delta \rrbracket = \max\{[\![b_1]\!], \ldots, [\![b_p]\!]\}$. This sequent is *valid* with respect to the interpretation $[\![\cdot]\!]$ if $\llbracket \Gamma \rrbracket \leqslant \llbracket \Delta \rrbracket$ holds. On the other hand, a *counter-model* to this sequent is an interpretation $[\![\cdot]\!]$ such that $\llbracket \Delta \rrbracket < \llbracket \Gamma \rrbracket$, i.e. for any pair (i,j), the inequality $[\![b_j]\!] < [\![a_i]\!]$ holds.

Proposition 1. *The sequent $a_1, \ldots, a_n \vdash b_1, \ldots, b_p$ has the same counter-models as the formula $(a_1 \wedge \cdots \wedge a_n) \supset (b_1 \vee \cdots \vee b_p)$.*

The proof of this proposition is trivial. Let f be a propositional formula. It can either be viewed as a boolean formula, or a formula of LC_n. We relate the two semantic interpretations using the double negation. We define the mappings $\phi : \{0,1\} \to \overline{[0,n)}$ and $\psi : \overline{[0,n)} \to \{0,1\}$ by $\phi_0 = 0$, $\phi_1 = \infty$, $\psi_0 = 0$ and $\psi_x = \infty$ for $x > 0$.

Proposition 2. *Let f be a propositional formula and $\sigma : \mathsf{LVar} \to \overline{[0,n)}$ be an interpretation of variables. The identity $[\![\neg\neg f]\!](\sigma) = \phi\big([\![f]\!](\psi \circ \sigma)\big)$ holds.*

Proof. Let us denote $\neg\neg f$ by $f^\star$. We remark that ϕ commutes with the interpretation of the connectives $\wedge$, $\vee$ and $\supset$; for instance $\max(\phi_x, \phi_y) = \phi(x \vee y)$. Then, the following logical identities hold in LC_n:

$$\begin{array}{ll} [\![\bot^\star]\!]_\sigma = [\![\bot]\!]_\sigma & [\![(a \vee b)^\star]\!]_\sigma = \max([\![a^\star]\!]_\sigma, [\![b^\star]\!]_\sigma) \\ [\![(a \supset b)^\star]\!]_\sigma = [\![a^\star]\!]_\sigma \twoheadrightarrow [\![b^\star]\!]_\sigma & [\![(a \wedge b)^\star]\!]_\sigma = \min([\![a^\star]\!]_\sigma, [\![b^\star]\!]_\sigma) \end{array}$$

[3] With the convention $\overline{[0,\infty)} = \mathbb{N} \cup \{\infty\}$. With our particular representation, the algebraic models $\overline{[0,n)}$ form a strictly increasing sequence of subsets of $\overline{\mathbb{N}}$.

We prove $[\![f^\star]\!](\sigma) = \phi\big([\![f]\!](\psi \circ \sigma)\big)$ by induction on f:

- if $f \equiv \bot$, we obtain $[\![\bot^\star]\!]_\sigma = [\![\bot]\!]_\sigma = 0$ and $\phi([\![\bot]\!](\psi \circ \sigma)) = 0$;
- if $f \equiv x$, then $[\![x^\star]\!]_\sigma = 0$ if $\sigma_x = 0$ and $[\![x^\star]\!]_\sigma = \infty$ if $\sigma_x > 0$. So $[\![x^\star]\!]_\sigma = \phi(\psi(\sigma_x))$. On the other hand, $\phi([\![x]\!](\psi \circ \sigma)) = \phi(\psi \circ \sigma(x)) = \phi(\psi(\sigma_x))$;
- if $f \equiv a \vee b$, then $[\![(a \vee b)^\star]\!]_\sigma = [\![a^\star \vee b^\star]\!]_\sigma = \max(\phi([\![a]\!](\psi \circ \sigma)), \phi([\![b]\!](\psi \circ \sigma))) = \phi(\max([\![a]\!](\psi \circ \sigma), [\![b]\!](\psi \circ \sigma))) = \phi([\![a \vee b]\!](\psi \circ \sigma))$;
- the cases $f \equiv a \wedge b$ and $f \equiv a \supset b$ are similar. □

5.2 Transforming conditional graphs into formulae

Let us consider a conditional graph $\mathcal{G}$. There may be some conditional formulae on green arrows like $\rightarrow_f$. We consider all these formulae and the propositional variables they contain. For each vertex v of $\mathcal{G}$ we introduce a new propositional variable x_v, so that the propositional variables occurring in conditional formulae and the newly introduced x_v do not overlap. We build a sequent $\mathcal{S}_\mathcal{G} = \Gamma \vdash \Delta$ by adding one formula to either Γ or Δ for each arrow of $\mathcal{G}$:

- if $v \rightarrow w$ is green arrow of $\mathcal{G}$, we add $x_v \supset x_w$ to Γ;
- if $v \rightarrow_f w$ is conditional green arrow of $\mathcal{G}$, we add $(\neg\neg f) \supset (x_v \supset x_w)$ to Γ;
- if $v \Rightarrow w$ is a red arrow of $\mathcal{G}$, we add $x_w \supset x_v$ to Δ.

Theorem 3. *Let $\mathcal{G}$ be a conditional graph, $\mathcal{S}_\mathcal{G}$ its associated sequent and $n \in \mathbb{N}$. There exists a valuation $\sigma : \mathsf{LVar} \rightarrow \{0,1\}$ such that the instance graph $\mathcal{G}_\sigma$ does not contain a $(n+1)$-alternating chain if and only if the sequent $\mathcal{S}_\mathcal{G}$ has a counter-model in LC_n.*

Proof. First, let us suppose that there exists a valuation $\sigma : \mathsf{LVar} \rightarrow \{0,1\}$ s.t. the instance graph $\mathcal{G}_\sigma$ does not contain a chain of type $(\rightarrow^\star\Rightarrow)^{n+1}$. By theorem 4 of [11], there exists a height h s.t. for every vertices v, w of $\mathcal{G}$ (or $\mathcal{G}_\sigma$, they have the same vertices), $h_v \in [0, n]$, if $v \rightarrow w \in \mathcal{G}_\sigma$ then $h_v \leqslant h_w$ and if $v \Rightarrow w \in \mathcal{G}_\sigma$ then $h_v < h_w$. We define $h'_v = n \twoheadrightarrow v$, i.e. $h'_v = h_v$ if $h_v < n$ and $h'_v = \infty$ if $h_v = n$. We define a valuation $\sigma' : \mathsf{LVar} \rightarrow \overline{[0, n)}$. If $x \equiv x_v$ is a variable corresponding to one of the vertices of $\mathcal{G}$ then $\sigma'_{x_v} = h'_v$. Otherwise we define $\sigma'_x = \phi(\sigma_x)$. We recall that the x_v do not overlap with the variables occurring in the conditional formulae of $\mathcal{G}$. We prove that σ' is a counter-model of $\mathcal{S}_\mathcal{G} \equiv \Gamma \vdash \Delta$:

- consider the formula $x_v \supset x_w$ occurring in Γ. Then $[\![x_v \supset x_w]\!]_{\sigma'} = [\![x_v]\!]_{\sigma'} \twoheadrightarrow [\![x_w]\!]_{\sigma'} = \sigma'(x_v) \twoheadrightarrow \sigma'(x_w) = h'_v \twoheadrightarrow h'_w = \infty$ because $h'_v \leqslant h'_w$ since $v \rightarrow w$ occurs in $\mathcal{G}_\sigma$. Thus $[\![x_v \supset x_w]\!]_{\sigma'} = \infty$;
- consider the formula $(\neg\neg f) \supset (x_v \supset x_w)$ occurring in Γ. Then $[\![(\neg\neg f) \supset (x_v \supset x_w)]\!]_{\sigma'} = [\![\neg\neg f]\!]_{\sigma'} \twoheadrightarrow [\![x_v \supset x_w]\!]_{\sigma'}$. $[\![\neg\neg f]\!]_{\sigma'} = \phi([\![f]\!](\psi \circ \sigma')) = \phi([\![f]\!](\psi \circ \phi \circ \sigma)) = \phi([\![f]\!]_\sigma)$. If $[\![f]\!]_\sigma = 0$ and in this case $[\![\neg\neg f]\!]_{\sigma'} \twoheadrightarrow [\![x_v \supset x_w]\!]_{\sigma'} = 0 \twoheadrightarrow [\![x_v \supset x_w]\!]_{\sigma'} = \infty$. On the other hand, if $[\![f]\!]_\sigma = 1$, then $v \rightarrow w$ occurs in $\mathcal{G}_\sigma$, and thus $h'_v \leqslant h'_w$. So $[\![\neg\neg f]\!]_{\sigma'} \twoheadrightarrow [\![x_v \supset x_w]\!]_{\sigma'} = \infty \twoheadrightarrow [\![x_v \supset x_w]\!]_{\sigma'} = [\![x_v \supset x_w]\!]_{\sigma'} = h'_v \twoheadrightarrow h'_w = \infty$. In either case, $[\![(\neg\neg f) \supset (x_v \supset x_w)]\!]_{\sigma'} = \infty$;

– consider the formula $x_w \supset x_v$ occurring in Δ. Then $v \Rightarrow w$ occurs in $\mathcal{G}_\sigma$. $[\![x_w \supset x_v]\!]_{\sigma'} = h'_w \twoheadrightarrow h'_v = h'_v$ since $h'_v < h'_w$. Moreover, as $h'_v < h'_w$, we deduce $h'_v < \infty$ and then $[\![x_w \supset x_v]\!]_{\sigma'} < \infty$.

Then $[\![\Delta]\!]_{\sigma'} < \infty$ and that $[\![\Gamma]\!]_{\sigma'} = \infty$. So σ' is a counter-model of $\mathcal{S}_\mathcal{G}$.

Conversely, consider $\sigma : \mathsf{LVar} \to \overline{[0,n)}$ a counter-model of $\mathcal{S}_\mathcal{G}$. We define $\sigma' = \psi \circ \sigma$. Let us consider a conditional arrow $v \to_f w$ of $\mathcal{G}$. We compute $[\![\neg\neg f]\!]_\sigma = \phi([\![f]\!]_{\sigma'})$. $[\![f]\!]_{\sigma'} = 1$ if and only if there is an arrow $v \to w$ in $\mathcal{G}_{\sigma'}$. If $[\![f]\!]_{\sigma'} = 1$ then $\phi([\![f]\!]_{\sigma'}) = \infty$ and $[\![(\neg\neg f) \supset (x_v \supset x_w)]\!]_\sigma = \infty \twoheadrightarrow [\![x_v \supset x_w]\!]_\sigma = [\![x_v \supset x_w]\!]_\sigma$. On the other hand, if $[\![f]\!]_{\sigma'} = 0$, then $[\![(\neg\neg f) \supset (x_v \supset x_w)]\!]_\sigma = 0 \twoheadrightarrow [\![x_v \supset x_w]\!]_\sigma = \infty$. Let use denote Γ' the multiset where we have replaced the formula $(\neg\neg f) \supset (x_v \supset x_w)$ by $x_v \supset x_w$ when $[\![f]\!]_{\sigma'} = 1$ and by nothing (i.e. we simply erase it) when $[\![f]\!]_{\sigma'} = 0$. It is clear that $[\![\Gamma']\!]_\sigma = [\![\Gamma]\!]_\sigma$. So σ is also a counter-model in LC_n of the implicational sequent $\Gamma' \vdash \Delta$ corresponding to the instance graph $\mathcal{G}_{\sigma'}$. According to theorem 6 of [11], since the implicational sequent $\Gamma' \vdash \Delta$ has a counter-model in LC_n, its instance graph $\mathcal{G}_{\sigma'}$ does not contain a chain of type $(\to^\star \Rightarrow)^{n+1}$. □

It is obvious that the size of $\mathcal{S}_\mathcal{G}$ is linear in the size of $\mathcal{G}$ (of course, the size of $\mathcal{G}$ should account for the size of conditional formulae). So there exists a linear transformation of the problem of resource consumption bounding into the decision problem in LC_n.

Corollary 2. *Let $\mathcal{E}$ be a normal set of process equations of size k and $\mathcal{V}$ the set of variables of $\mathcal{E}$. There exists a formula $f_\mathcal{E}$ of LC of size $\mathcal{O}(k)$ such that for any $n \in \mathbb{N}$, $(\mathcal{E}, \mathcal{V})$ has a resource use boundable by n if and only if $f_\mathcal{E}$ has a counter-model in LC_n.*

Proof. Let $\mathcal{G}$ be the conditional graph associated to $\mathcal{E}$ (see corollary 1), $\mathcal{S}$ be the sequent associated to $\mathcal{G}$ (see theorem 3). Then $f_\mathcal{E}$ is a formula logically equivalent to $\mathcal{S}$, see proposition 1. □

6 Counter-models of LC_n and resource consumption

In this section, we briefly recall some of our previous results which provide the proof of the converse of corollary 2. Given a formula f of LC, we explain how to build a conditional graph $\mathcal{G}_f$ of size linear w.r.t. the size of f which represents the proof-search process on f and on which it is possible to either prove f or extract counter-models of f. For an explanation of the construction in full details, the reader is invited to consult [12].

Let f be a formula of LC. We build a conditional graph $\mathcal{G}_f$ based on the decomposition tree of f, i.e. the set of sub-formula occurrences of f. An occurrence of a sub-formula r can be identified with the root node of the sub-tree corresponding to r. Each node is *polarized* starting with polarity $-$ for the root f^- and propagated the following way: the connectives $\vee$ and $\wedge$ preserve the polarity while the connective $\supset$ preserves the polarity on the right hand side and inverses it on the left-hand side.

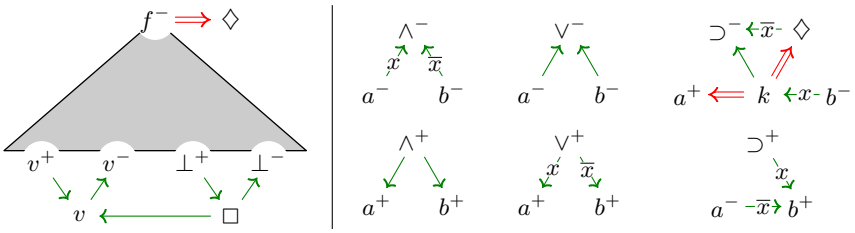


Fig. 2. Counter-model search system for LC

We add a supplementary node for each propositional variable v occurring in f. This is one node per variable, not per occurrence: two occurrences of the same variable only produce one supplementary node. These added nodes are not polarized. Two more nodes are added: $\Diamond$ and $\Box$. Intuitively, $\Diamond$ represents the semantic value ∞ whereas $\Box$ represents the semantic value 0.

Before we describe how we build the arrows of $\mathcal{G}_f$, we precise that we will sometimes introduce new conditional variables denoted x and use either x or its negation $\overline{x} \equiv \neg x$ as a condition indexing some green arrows like $\rightarrow_x$ or $\rightarrow_{\overline{x}}$. Now we describe how we build the arrows of $\mathcal{G}_f$:

- for the root f^-, we add a red arrows $f \Rightarrow \Diamond$;
- for every added node v corresponding to a propositional variable occurring in f, we add a green arrow $\Box \rightarrow v$;
- for a positive occurrence v^+ of a variable in f, we add a green arrow $v^+ \rightarrow v$;
- for a negative occurrence v^- of a variable in f, we add a green arrow $v \rightarrow v^-$;
- for a positive occurrence of $\bot^+$ in f, we add a green arrow $\bot^+ \rightarrow \Box$;
- for a negative occurrence of $\bot^-$ in f, we add a green arrow $\Box \rightarrow \bot^-$.

All these rules correspond to the left part of figure 2. Now we describe what we do with internal nodes, which corresponds to the right part of figure 2:

- for each negative occurrence of a sub-formula $r^- \equiv a^- \wedge b^-$, we introduce a new conditional variable x and we add the two conditional green arrows $a^- \rightarrow_x r^-$ and $b^- \rightarrow_{\overline{x}} r^-$;
- for each positive occurrence of a sub-formula $r^+ \equiv a^+ \wedge b^+$, we add the two green arrows $r^+ \rightarrow a^+$ and $r^+ \rightarrow b^+$;
- for $r^- \equiv a^- \vee b^-$, we add the two green arrows $a^- \rightarrow r^-$ and $b^- \rightarrow r^-$;
- for $r^+ \equiv a^+ \vee b^+$, we introduce a new variable x and add the two arrows $r^+ \rightarrow_x a^+$ and $r^+ \rightarrow_{\overline{x}} b^+$;
- for $r^- \equiv a^+ \supset b^-$, we introduce a new variable x and a new node k and the five arrows $b^- \rightarrow_x k$, $k \Rightarrow a^+$, $k \rightarrow r^-$, $k \Rightarrow \Diamond$ and $\Diamond \rightarrow_{\overline{x}} r^-$;
- for $r^+ \equiv a^- \supset b^+$, we introduce a variable new x and add the two arrows $r^+ \rightarrow_x b^+$ and $a^- \rightarrow_{\overline{x}} b^+$.

The construction of $\mathcal{G}_f$ is finished after each internal node has been processed. The order in which they are processed is indifferent. We recall the main result[4] of [11], which relates conditional graphs and LC_n:

Theorem 4. *Let f be a formula of LC, $\mathcal{G} \equiv \mathcal{G}_f$ be the conditional graph associated to f by the construction described in this section and $n \in \mathbb{N}$. Then f has a counter-model in LC_n if and only if there exists a valuation σ such that the instance graph $\mathcal{G}_\sigma$ has no $(n+1)$-alternating chain.*

Corollary 3. *Let f be a formula of LC of size k. There exists a process system $(\mathcal{E}, \mathcal{V})$ of size $\mathcal{O}(k)$ such that for any $n \in \mathbb{N}$, $(\mathcal{E}, \mathcal{V})$ has resource use boundable by n if and only if f has a counter-model in LC_n.*

Proof. Let f be a formula of LC. We apply theorem 4 and obtain a conditional graph $\mathcal{G}_f$. The size of $\mathcal{G}_f$ is linear w.r.t. the size of f. Then we apply theorem 2 and obtain a process system $(\mathcal{E}, \mathcal{V})$. The size of $(\mathcal{E}, \mathcal{V})$ is linear w.r.t. the size of $\mathcal{G}_f$, thus also w.r.t. the size of f. $(\mathcal{E}, \mathcal{V})$ has resource use boundable by n if and only if $\mathcal{G}_f$ has no $(n+1)$-alternating chains if and only if f has a counter-model in LC_n. □

7 Conclusion

We have defined a process calculus and an operational semantics that measures resource consumption. We establish a correspondence between normal process systems and conditional bi-colored graphs: a process system has resource use boundable by an integer n if and only if the corresponding graph has no $(n+1)$-alternating chain. Then we show how the absence of $(n+1)$-alternating chain in a graph can be expressed by the refutability of a formula of LC_n. Combining the two results, we get a linear transformation of a resource bounding problem in a process calculus into a decision problem in LC_n.

We recall our previous result on counter-model search in LC [11] and relate it to the process formalism we introduced. Thus, we have a linear transformation of a decision problem in LC_n into a resource bounding problem in a process calculus. This establishes the linear equivalence of the two problems. So LC could be viewed as a logic for specifying some resource related properties. This sheds some new lights on LC.

In further studies, we want to use the process calculus defined in this paper as an abstraction of more complex calculi: by keeping only the basic constructs present in our calculus and abstracting the other constructs. Counter-model search in LC could be used as a tool to bound resource consumption in some specified complex systems.

[4] To be precise, the construction of $\mathcal{G}_f$ we present here, and the associated theorem 4 are a slight but obvious modification of the cited result, to integrate the case of the $\bot$ logical constant and to avoid generating conditional red arrows like $\Rightarrow_x$ or $\Rightarrow_{\overline{x}}$.

References

1. Gödel, K.: Zum intuitionistischen Aussagenkalkül. In: Anzeiger Akademie des Wissenschaften Wien. Volume 69. (1932) 65–66
2. Dummett, M.: A Propositional Calculus with a Denumerable matrix. Journal of Symbolic Logic **24** (1959) 96–107
3. Hajek, P.: Metamathematics of Fuzzy Logic. Kluwer Academic Publishers (1998)
4. Dyckhoff, R.: Contraction-free Sequent Calculi for Intuitionistic Logic. Journal of Symbolic Logic **57** (1992) 795–807
5. Dyckhoff, R.: A Deterministic Terminating Sequent Calculus for Gödel-Dummett logic. Logical Journal of the IGPL **7** (1999) 319–326
6. Avellone, A., Ferrari, M., Miglioli, P.: Duplication-Free Tableau Calculi and Related Cut-Free Sequent Calculi for the Interpolable Propositional Intermediate Logics. Logic Journal of the IGPL **7** (1999) 447–480
7. Avron, A.: A Tableau System for Gödel-Dummett Logic Based on a Hypersequent Calculus. In: TABLEAUX 2000. Volume 1847 of LNAI. (2000) 98–111
8. Metcalfe, G., Olivetti, N., Gabbay, D.: Goal-Directed Calculi for Gödel-Dummett Logics. In: CSL. Volume 2803 of LNCS. (2003) 413–426
9. Larchey-Wendling, D.: Combining Proof-Search and Counter-Model Construction for Deciding Gödel-Dummett Logic. In: CADE-18. Volume 2392 of LNAI. (2002) 94–110
10. Baaz, M., Ciabattoni, A., Fermüller, C.: Hypersequent Calculi for Gödel Logics – A Survey. Journal of Logic and Computation **13** (2003) 835–861
11. Larchey-Wendling, D.: Counter-model search in Gödel-Dummett logics. In: IJCAR 2004. Volume 3097 of LNAI. (2004) 274–288
12. Larchey-Wendling, D.: Gödel-Dummett counter-models through matrix computation. Electronic Notes in Theoretical Computer Science **125** (2005) 137–148
13. Milner, R.: Communication and Concurrency. International series in computer science. Prentice Hall (1989)